APP 5 : gestion des exceptions

Objectifs :

- Définir c'est quoi une exception et pourquoi elle se produit.
- Expliquer les avantages de l'utilisation de la gestion des exceptions Java
- Faire la distinction entre les exceptions capturées et non-capturées.
- Ecrire une simple routine de gestion des exceptions pour gérer une exception en utilisant les blocs *try–catch* et *try–final.*
- Ecrire des méthodes qui utilisent les déclarations *throws* pour la gestion des exceptions.

Activité 1: La génération des exceptions

1) Tester ces différents cas et lisez les messages qui seront affichés au niveau de la console de votre IDE:

   1er cas :

```
public class SuperHotel {
    static int x[];


    public void reserver(){

              x[0] = 1;
    }

    public static void main(String args[]) {

        SuperHotel s = new SuperHotel();
        s.reserver();
    }
}
```

2eme cas :

NB : Testez le cas si l'utilisateur saisi 0

```
public class DivisionExecption {


      static  int x =20;
      static int y ;
           public static void main(String[] args) {
          Scanner scanner = new Scanner(System.in);
             System.out.println(" Enter un entier");
              y = scanner.nextInt();
             System.out.println(x/y);
     }


}
```

3eme cas :

```
       public class AppelMethod {

                public void method1(){
                this.method2();
            }

            public void method2(){
               this.method1();
            }

           public static void main(String[] args) {

                AppelMethod appel = new AppelMethod();
                          appel.method1();
       }


       }
```

4eme cas :

```
public class AddTable {
    public static void main(String[] args) {

                int[] array = new int[3];
                for(int i=0;i<4;++i){
                        array[i] = i;
                }
                System.out.println(array);
        }
}
```

2) Donner une solution afin de résoudre ces problèmes (voir le cours)

Activité 2: La gestion des exceptions en utilisant le bloc try-catch

Classe : TestExceptions.java.

1. Essayez de compiler le fichier TestExceptions.java, que remarquez-vous?
2. Modifier TestExceptions.java pour gérer l'exception **IOException** à l'aide d'un bloc try-catch au lieu de propager l'exception. Une fois cette exception est déclenchée affichez la chaîne suivante: "The file you have requested cannot be found."
3. Pourquoi le programme compile maintenant lorsqu'on gère une des différentes exceptions listées?
4. Exécutez le programme, qu'est-ce que vous remarquez par rapport aux exceptions traités dans l'**Activité 1**?

Classe : MultiCatch.java

5. Corriger la classe MultiCatch.java

Activité 3: La Propagation des exceptions en utilisant throws

6. Modifier la déclaration de la méthode main() dans TestExceptions.java pour propager l'exception ***IOException*** *en utilisant le* ***throws***.
7. Compilez et exécutez le programme.

Activité 4: Le bloc *finally*

Classe TestFinally.java

1) Compilez et exécutez le fichier TestFinally.java.
2) Modifiez dans le fichier TestFinally.java afin de gérer l'exception en utilisant un bloc try-catch dans la méthode main(), en suivant les instructions mentionnés en commentaire

Activité 5: Création d'une exception personnalisé

Classe Event.java

Modifier la classe Event.java afin de déclencher une exception lorsque le nombre de participant atteint 100 participants et d'afficher le message « évènement complet ! »